The Design and Implementation of A Network Provenance System Framework

Xiang Gao
School of Computer Science, Northwestern Polytechnical University, Xi'an,China
Email: gaoxg@nwpu.edu.cn

Xiao Wang
School of Computer Science, Northwestern Polytechnical University, Xi'an,China
Email: wangxiao5018@163.com

Min Wang
School of communication, Air Force Engineering University, Xi'an, China
E-mail:wang_min5460@sina.com.cn

Yinghan Jin
Northwestern Polytechnical University, Xi'an,China
E-mail:248492720@qq.com

***Abstract*—Network forensic analysis and fault diagnosis are becoming increasingly important in network management and network security domain. This requires network management system has the ability to query network metadata, i.e. the network provenance functionality. For instance, network provenance can be used in tracking the path of dataflow through the network to obtain the source of message data.**

This paper presents the design and implementation of a network provenance system (NPS) framework, the framework is used to support the full range of functionality required for enabling forensics in distributed systems. We adopt the declarative networking coding method proposed in the networking domain to maintain and query distributed network provenance. The framework prototype is developed using Rapidnet, a declarative networking platform based on the ns-3 network simulator. Simulation experiments are conducted in simulated network, the experiment results indicates that our network provenance system could support provenance process in a large-scale distributed network and significantly reduce bandwidth cost compared to traditional approach.

***Index Terms*—Network Provenance, Declarative Networking, Query Customization**

I. INTRODUCTION

The network provenance is used for network metadata inquiry .Network provenance describes the history and derivations of network state when executing distributed protocol, it is widely deployed on thousands of nodes across multiple administrative domains and geographical areas.

Traditional data provenance technology has been extensively applied to a variety of areas, including probabilistic databases [1], collaborative databases [2], file systems, scientific workflow computation [3,4], and cloud computing[5].For instance, VisTrails[3] present solutions for historical provenance, it enables reproducibility and simplifies the complex problem of creating and maintaining visualization products. Orchestra[2],focuses on managing disagreement among multiple data representations, in which independent researchers or groups with different schemas and data can share information in the absence of global agreement. RAMP [5] discuss distributed provenance maintenance and querying for specific applications, recording metadata related to provenance or develop a logging mechanism for tracking information and dependencies between data. Sprov[6]enforces the integrity of chain-structured provenance, this further suggests a comprehensive provenance representation that uses polynomials.

To support the full range of functionality required for enabling forensics in distributed systems, there are a number of challenges that traditional data provenance can not answer very well. In this paper, we propose a new provenance approach integrate with declarative networking model, the main framework can be implemented compactly using declarative language.

The remainder of this paper is organized as follows:(1)We first present a background introduction to

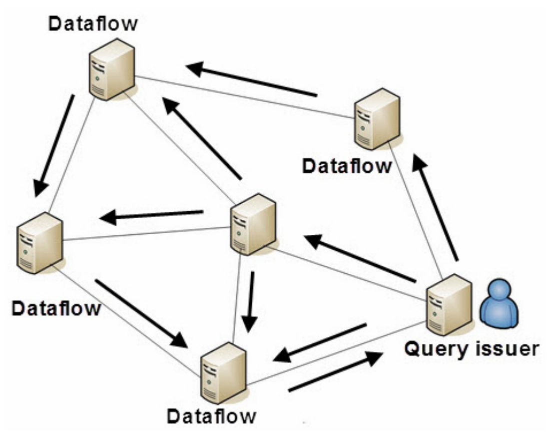
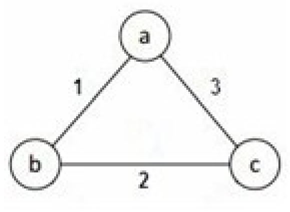


Figure1. Distributed dataflow between nodes



link(@source, destination, cost)		
source	destination	cost
a	b	1
a	c	3
b	a	1
b	c	2
c	a	3
c	b	2

Figure2. Network topology

declarative networking in section 2.(2)The framework use distributed relational tables to store provenance information. We propose a reference-based approach that reactively generates reverse markers that can be recursively traversed to a query. (3)The system consists of provenance maintenance module and provenance query module. Maintenance module generate and update the provenance data and query module accept provenance query request from users.(4)several modification and optimizations are made to prototype in order to further reduce the communication overhead among nodes and adapt specific queries under certain circumstance.(5) Contrast experiments are conducted on the simulated network are presented in Section 5.

II. Declarative Networking

The distributed nature and large scale of today's networks pose challenges for the Network protocol design and implementation, it is important to balance the extensibility and flexibility of network protocols while guaranteeing the robustness and efficiency as well. The most fundamental matter that network protocol designer faces is how to divorce from the tedious protocol implementation and focus on the functionality and concept design of protocols. Declarative networking language[7] allows us to specify at a high level "what to do", rather than "how to do it", it focus on solving the problems of traditional distributed programming and parallel programming, simplifying coding process and significantly reducing the size of implementation in lines of code.

A. NDlog Language

Declarative language include Datalog, Netlog, overlog, NDlog, in this paper we mainly use NDlog[8],which stands for Network Datalog.NDlog language can express a variety of well-known routing protocols (e.g., distance vector, path vector, dynamic source routing, link state, multicast) in a compact and clean fashion, typically in a handful of lines of program code.

Network protocols are expressed in NDlog and distributed among nodes, each node compile NDlog rules into distributed dataflow, as is shown in Fig. 1.These dataflow exchange messages and network state between nodes until fixpoint is reached.

TABLE I.
EXAMPLE PROV RELATION TABLE

Loc	VID	RID	RLoc	Derivation
a	VID5	RID2	a	pathCost(@a,c,3)
a	VID5	RID3	b	pathCost(@a,c,3)
a	VID7	RID5	a	minCost(@a,c,3)
b	VID4	RID1	b	pathCost(@b,c,2)
b	VID6	RID4	b	minCost(@b,c,2)

TABLE II.
EXAMPLE RULEEXEC RELATION TABLE

RLoc	RID	R	VIDList	Derivation
a	RID2	sp1	(VID3)	pathCost(@a,c,3)
a	RID5	sp3	(VID5)	minCost(@a,c,3)
b	RID1	sp1	(VID1)	pathCost(@b,c,2)
b	RID3	sp2	(VID2,VID6)	pathCost(@a,c,3)
b	RID4	sp3	(VID4)	minCost(@b,c,2)

A NDlog rule has the form p :− q_1,q_2...,q_n.The predicate p is the head of the rule, and q_1, q_2, ..., q_n constitutes the body of the rule. q_i are either predicates over fields or function applied to fields. Each predicate has a primary key, which contains a set of fields uniquely identifying each tuple. Only when all the functions and predicates are satisfied, the rule is executed and generates head. Consider the Mincost program shown below.

```
        sp1 pathCost(@S,D,C) :- link(@S,D,C).
        sp2 pathCost(@S,D,C1+C2): -
            link(@S,Z,C1),minCost(@Z,D,C2).
        sp3 minCost(@S,D,min<C>):-
            pathCost(@S,D,C).
```

The Mincost program executes on a example network shown in Fig. 2, each node is initialized with a link tuple for each of its neighbors. Rule sp1 generates one-hop pathCost tuple from local link relation table. Rule sp3 generates minCost tuple from locally stored pathCost table. To generate new pathCost tuple, rule sp2 use local link tuple to concatenate with minCost tuple generated in the previous round. NDlog prepend an "@" symbol to a single field, denoting the tuple's actual storage location. The location specifier enable the NDlog language with distributed computing capabilities.

B. Rapidnet Platform

RapidNet[9] is a development toolkit based on NS-3 simulator[10] for simulation and implementation of network protocols. The NS-3 network simulator is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. All simulations in NS-3 are driven by discrete event.NS-3 provides simulation and research for a variety of networks, protocols and layers, researchers can make any modifications according to their needs.

Rapidnet integrates declarative networking engine on NS-3 to execute declarative programs. In rapidnet, protocols are expressed in Ndlog code, the RapidNet compiler then parses the input NDlog protocols into C++ code that runs over the RapidNet Library. The latest Rapidnet version released is Rapidnet_v0.3.

III. Declarative Provenance Framework

The declarative provenance system is a efficient provenance framework used in distributed environment, it adopts distributed provenance and uses reference based ways to exchange information between nodes, resulting in significant reductions in bandwidth costs compared to traditional approaches.

A. Provenance Granularity

Three levels of granularity are provided to adjust the level of detail and influence on the network performance. (1)Tuple level provenance has the ability to trace a tuple's complete construction process, tuple level provenance includs all intermediate tuple in the derivation process, it plays an important role in the network protocol debugging.Tuple level provenance consists of all nodes and edges in the graph and encodes the maximum amount of information, which incurs the largest communication overhead. (2)Node level provenance encodes only the nodes that are involved in the derivation. It is used to determine which node of the network is responsible for a given tuple. (3)Provenance information may also stored at the trust domain level.All nodes in the trust domain share a domain identifier[11], provenance information encodes trust domain which involved in the derivation process.

B. Data Model

The provenance framework use directed acyclic graph G(V,E)[11] as data model of provenance information. Each edge in the edge set represents a rule execution between computation result and tuples. Edge from tuple to computation result represents the input of rules and edge from the computation results to tuple represents the output of rules. The vertex set includes tuple vertices and rule execution vertices, identified by VID and RID. To uniquely identify each tuple vertex in a derivation graph, we assign a vertex ID to each vertex in the provenance graph, using cryptographic hash functions[12] to generate a unique value. For instance, the VID of a tuple vertex for link(@X,Y,C) is VID =SHA1("link"+X+Y+C). For rule execution vertex, a rule ID has the form RID = SHA1("r2"+X+t1+t2),which means rule r2 at node X is executed with input tuples t1 and t2 .

C. Provenance Maintenance Module

The provenance information is maintained by two relational tables Prov and ruleExec,shown in table 1 and table 2.The Prov table stores provenance information, each entry represents the derivation of a given tuple, which has the form prov(@Loc,VID,RID,RLoc).The Prov table is distributed throughout the nodes by the location identifier Loc. The ruleExec table stores the metadata generated by rule execution, the RLoc field denotes the rule reside location, each entry represents a rule execution, which has the form ruleExec (@RLoc,

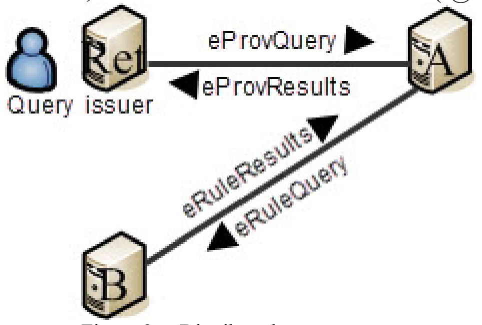


Figure 3. Distributed query process

RID,R,VIDList).

Two NDlog maintain rules are used to generate provenance information while executing network protocols. For instance, the provenance maintenance rules for Mincost rule sp2 are automatically rewritten in provenance generation rules.

```
R1 ruleExec(@RLoc,RID,R,List) :-
    ePathCostTemp(@RLoc,S,D,C,RID,R,List).
R2 prov(@S,VID,RID,RLoc):-
    ePathCost(@S,D,C,RID,RLoc),
    VID=f_sha1( ''pathCost'' +S+D+C).
```

The first three provenance generation rules generates the S,D and C fields that the Mincost protocol requires, besides, these three rules generates four maintenance fields: RLoc,RID,R and LIST. These four fields consist of the ruleExec entry in rule R1.Then,the ePathCost event tuple generated by rule R1 is sent to node S according to the location identifier. At node S, the received ePathCost event tuple is used to execute the protocol and generate the Prov entry (rule R2) .

D. Provenance Query Module

Consider the provenance query process of minCost tuple shown in table 1, the corresponding VID value of the tuple is VID7,a VID 7 query is initialized at first, the corresponding RID value RID5 is found according to Prov table(3rd entry).The query then trace back to the VIDList corresponding to VID5 in the ruleExec table(2nd entry) using fields RLoc and RID, and two qualified entry is returned form Prov table, triggering two VID5 query simultaneously(VID5 is obtained from ruleExec entry).The process is similar to the VID7 query. The VID5 query will trigger new VID query until fixpoint is reached. After all the children of VID5 have returned the Prov entry, the provenance information is

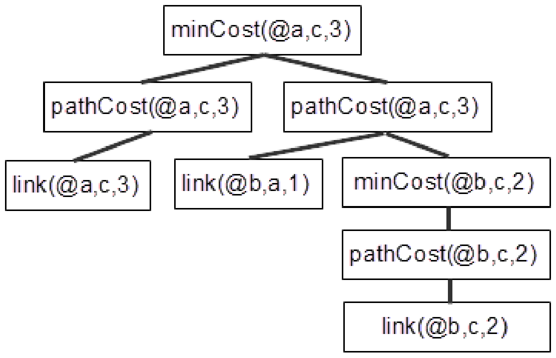


Figure 4. Provenance graph for minCost(@a,c,5)

sent to the query issuer by RLoc node.

Above is the general process of the provenance information query, the query module implementation is achieved by two basic rules (rule edb1 and c0) and eight recursive rules(idb1-idb4,rv1-rv4).Rule edb1 return basic tuple provenance information, rule c0 count the a certain VID's children number. There are two pairs of query/response event tuple (eProvQuery/eProvResults and eRuleQuery/ eRuleResults) operating Prov table and ruleExec table, respectively.

The idb rules(idb1-idb4) is used to operate Prov table, the Request node initialize a VID query request eProvQuery(@X,QID,VID,Ret), where QID is the query ID uniquely differ from other queries. If the VID being queried is a base tuple, the module execute rule edb1 to end this query, otherwise, rule idb1 initialize provenance buffer at the given address provided by tuple eProvQuery. Rule idb2 send event tuple eRuleQuery to node RLoc(protocol execution node), retrievaling qualified entry according to field RID. The rv rule (rv1-rv4) is used to trace rule execution vertices in the ruleExec table, performing a similar traversal as idb rules did.

Idb rules and rv rules are triggered recursively until all child nodes are visited. As the query proceeds, the eprovQuery events tuples are recursively propagated from the query issuer towards the base tuple in order to construct the entire provenance graph. Finally, the provenance query module recursively traverses prov and ruleExec tables across nodes until the entire provenance graph is obtained.

IV. Prototype Optimization

A. Query Traversal Order

During the provenance query stage, the framework prototype issues queries simultaneously to all possible directions, the recursive queries use breadth first order to traverse all the vertex of current layer in the provenance tree. As a result, BFS would visit all the vertex throughout the whole provenance tree, each level of the tree is visited in parallel at different nodes.

In this section, We explore another querying traversal order. Instead of starting queries for each derivation simultaneously, in depth first order(DFS), vertices are visited along the tuple's derivation direction until it reached the base tuple vertex.The processing of the next derivation is started only if the results of the previous direction have been received. DFS will terminates such queries as soon as certain conditions are met (e.g.,the given length of derivation is obtained or the derivation number of certain tuple is recieved).DFS is used in queries which the users want to konw whether a tuple has more than N derivations or the length of certain derivatin of a tuple. However, the new approach may incur longer querying latencies than the prototype since the former order could stall before a subresult of a derivation is obtained,as is shown in Fig. 4.

```
idb1 pResultTmp(@X,QID,Ret,VID,f_empty()):-
        eProvQuery(@X,QID,VID,Ret),
        prov(@X,VID,RID,RLoc),RID!=NULL.
idb2a pQList(@X,QID,AGGLIST<RID,RLoc>) :-
        eProvQuery(@X,QID,UID,Ret),
        prov(@X,UID,RID,RLoc), RID!=NULL.
idb2b eIterate(@X,QID,N) :-
        pResultTmp(@X,QID,Ret,UID,Buf),
         numChild(@X,UID,C),N=f_size(Buf)+1,
         N<=C,f_pIDB(Buf,X)<=Threshold.
idb2c eRuleQuery(@RLoc,RQID,RID,X) :-
        eIterate(@X,QID,N),
        pQList(@X,QID,LRID,LRLoc),
        RID=f_item(LRID), RLoc=f_item(LRLoc).
```

The Request node initialize a VID query request eProvQuery(@X,QID,VID,Ret),where QID is the query ID uniquely differ from other queries. If the VID being queried is a base tuple, the module execute rule edb1 to end this query, otherwise, rule idb1 initialize provenance buffer at the given address provided by tuple eProvQuery.Rule idb2 send event tuple eRuleQuery to

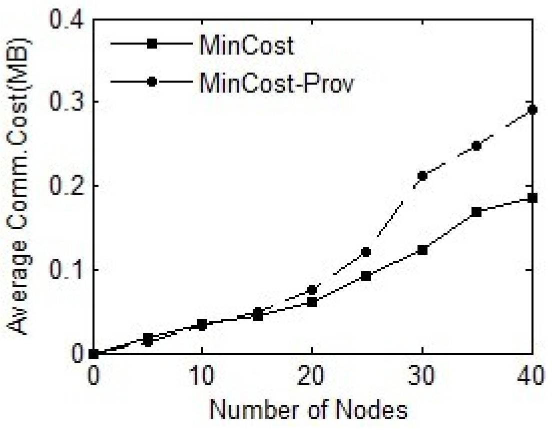


Figure 5 .Average communication cost(MB) for provenance maintenance

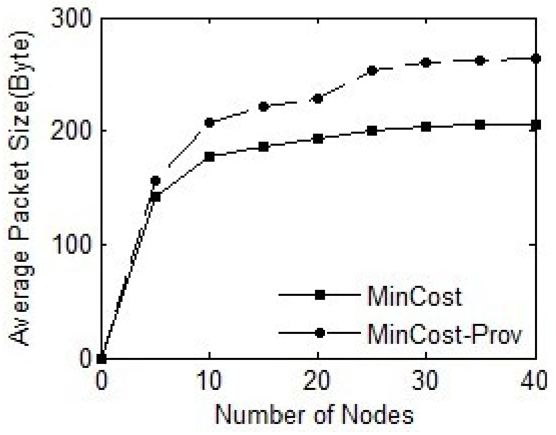


Figure 6 .Average packet size(Byte) for provenance maintenance

node RLoc(protocol execution node), retrievaling qualified entry according to field RID. To allow DFS terminates queries as soon as the query requirement is satisfied, the original idb2 rule is rewritten by three rules (idb2a,idb2b and idb2c) . Rule idb2a maintains a pOList which stores the rule execution vertices while rule idb2b counts the number of the results that have been ever received by checking the size of the buffer. If the condition is not met (e.g., derivation number is not reached), query proceed to visit the next derivation

B. *Provenance Condensation*

In the framework prototype, three user-defined functions are used in the query rules to return provenance information. These user-defined functions are implemented as (1)f_pEDB(VID),it takes as input the base tuple's VID. The function returns the base tuple or the relevant Prov entry.(2)f_pIDB(Derivations, Loc),it takes as input Derivations that contain all possible ways to derive the tuple, and the location specifier of the tuple. The function applies a union operation across all entries in Derivations.(3) f_pRule(ChildPred,R,RLoc),it takes as input ChildPred, representing all input tuples that are used in the execution of rule R at location Loc. The function applies a join operation across all entries in ChildPred .

To further reduce the bandwidth cost between nodes, the f_pEDB function can be modified to return the VID value instead of the whole Prov entry. The returned VID value could then be retrieved by reading a local Prov table that maps VIDs to tuples. The second approach tries to calculate the derivation number of a given tuple. For base tuple, f_pEDB returns a integer, representing a edb tuple. For intermediate tuple, f_pIDB returns the sum of the sub-results; f_pRULE returns the product of the sub-results. Such functions are used in scenarios which users want to obtain the derivation number of a given tuple.

V. Evaluation

Our experiments are conducted on simulated networks. We use the NS-3 network simulator developed by the University of California Berkeley to do the simulation experiments. In exploration and validation of section 4, we use the 3.5.1 version of NS-3 to build up network topological structure and Rapidnet complier to compile NDlog programs under Ubuntu10.14.

A. *Prototype Analysis*

We compare two implementations of the MinCost routing protocol: MinCost is a base implementation in Rapidnet without provenance and MinCost-Prov is an implementation of the declarative provenance prototype.

Before each execution, each node is initialized with a link tuple for each of its neighbors, and the experiment terminates until no more tuples can be derived.

We conduct the experiment in multiple node size, arranging from 5 to 40.Fig. 5 plots the average communication overhead per-node and Fig. 6 plots the average packet size during provenance maintenance stage. MinCost-Prov incurs more communication overhead and larger packet size than MinCost, that is because the MinCost-Prov attach additional provenance in each tuple. Each tuple have to carries its derivation history when exchange between nodes.

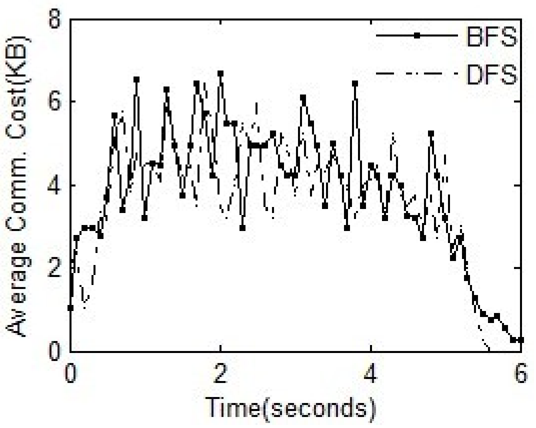


Figure 7. Average communication cost(KB) for different query traversal order in a 30-node network

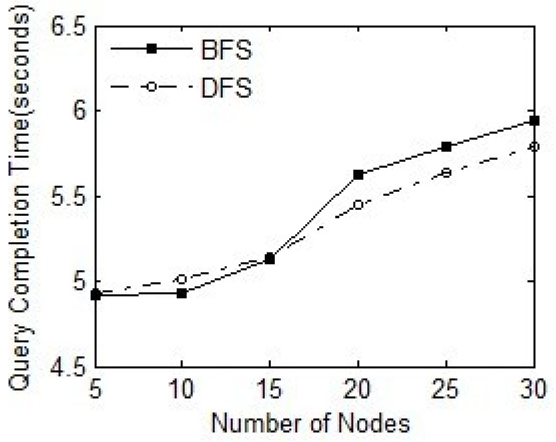


Figure 8 .Average Completion time(seconds) for different query traversal order in various node size

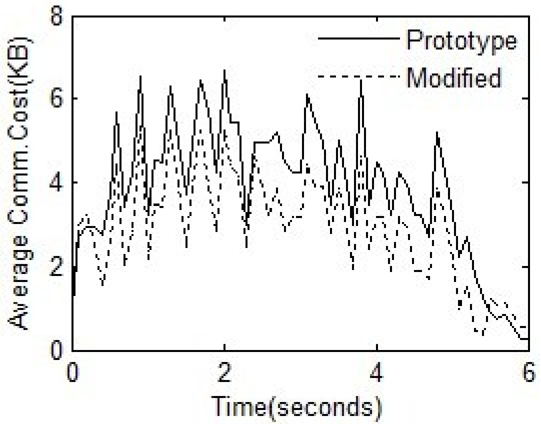


Figure 9. Average communication cost(KB) for different querying granularity in a 30-node network

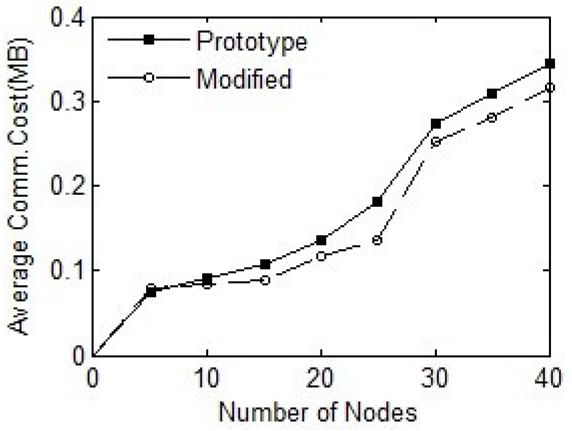


Figure 10.Average communication cost(MB) for different querying granularity in various node size

B. *Modified Framework Evaluation*

The traversal experiment is conducted in different network size (number of nodes).To study the trade-offs between different query traversal orders,we compare the average communication cost of these two methods to verify our analysis in section 4.1.

Two different provenance traversal order are being tested in various node size. Breath first traversal order's query completion time is mainly determined by traversal depth in the provenance graph. Depth first traversal order visit alternative derivation directions according to the given target function. Fig. 7 plots the average communication cost using depth first traversal order in a 30-node size and Fig. 8 shows the average convergence time ranging from 5 nodes to 40 nodes. The experiment result indicates that in conditional queries, the depth first order has the approximately equivalent communication cost compared to breath first order but converges quicker than breath first order traversal.

We conduct the provenance condensation of node-level granularity on Rapidnet platform, modified functions are added in the rapidnet source file.

Fig. 9 shows the average communication cost of provenance query of modified approach in a 30-node network during 6 seconds. Fig. 10 plots the average communication cost in various node size. The simulation result indicate that due to the prov entry condensation, the provenance condensation approach further reduce the average communication cost compared to prototype implementation.

VI. Conclusion

This paper presents a generic efficient provenance framework used in distributed environment, it adopts distributed provenance and uses reference based ways to exchange information between nodes, resulting in significant reductions in bandwidth costs compared to traditional approaches. Administrators can explain any network state changes using this provenance framework, it is used for provenance and forensics against illegal behaviors in network. Our prototype use graph representation to express provenance information and two relational tables to store provenance data. Five maintenance rules in maintenance module generate and update the provenance data, eight query rules in query module accept provenance query request from users.

In the exploration and validation process, we use the 3.5.1 version of NS-3 to build up network topological structure and Rapidnet complier to compile NDlog programs under Ubuntu10.14.The prototype analysis indicates that declarative provenance system effectively reduce the communication overhead while ensuring efficient data provenance compared to other approaches. The query customization evaluation shows that depth first order traversal is more efficient in queries which the users want to know whether a tuple has more than N derivations or the length of certain derivation of a tuple. And provenance condensation incurs lower communication cost while reducing query completion time. In future work, we will explore building efficient provenance framework in adversarial environment. Consider the integration of the provenance framework with existing fault detection method[13] to ensure the confidentiality and authenticity of provenance information.

ACKNOWLEDGMENTS

This work is supported by National Key Technology R&D Program of the Ministry of Science and Technology（2012BAB15B01）

REFERENCE

[1] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance.In *Proceedings of the International Conference on Database Theory (ICDT)*, 2001.

[2] Todd J. Green, Grigoris Karvounarakis, Nicholas E. Taylor, Olivier Biton,Zachary G. Ives, and Val Tannen. ORCHESTRA: Facilitating collaborative data sharing. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2007.

[3] Steven Callahan, Juliana Freire, Emanuele Santos, Carlos Scheidegger, Clau-dio Silva, and Huy Vo. VisTrails: Visualization meets data management. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2006.

[4] Sarah Cohen-Boulakia, Olivier Biton, Shirley Cohen, and Susan Davidson.Addressing the provenance challenge using zoom.*Concurrency and Computation : Practice and Experience*, 20:497–506, 2008.

[5] Robert Ikeda, Hyunjung Park, and Jennifer Widom. Provenance for generalized map and reduce workflows.In *Proceedings of Biennial Conference on Innovative Data System Research (CIDR)*, 2011.

[6] Ragib Hasan, Radu Sion, and Marianne Winslett. Preventing history forgery with secure provenance.*ACM Transactions on Storage (TOS)*, 5(4):1–43, 2009.

[7] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M.Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative Networking. In *CACM*, 2009.

[8] B. T. Loo, T. Condie, J. M. Hellerstein, I. Stoica and R. Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *SIGCOMM*, 2005.

[9] RapidNet.http://netdb.cis.upenn.edu/rapidnet/.

[10] Network Simulator 3.http://www.nsnam.org/.

[11] W. Zhou, M. Sherr, T. Tao, X. Li, B. T. Loo, and Y. Mao.Efficient querying and maintenance of network provenance at internet-scale. In *SIGMOD*, 2010.

[12] Hua Zheng,Qing hua Zhu,Kewen Wu.Provenance Management for Data Quality Assessment.In *Journal of Software*, Vol 7, No 8 (2012), 1905-1910, Aug 2012

[13] Wei Li, Dawu Gu, Xiaoling Xia, Ya Liu, Zhiqiang Liu.Fault Detection on the Software Implementation of CLEFIA Lightweight Cipher.In *Journal of Networks*, Vol 7, No 8 (2012), 1288-1294, Aug 2012

Gao Xiang was born in ShangHai, China, in April 10, 1974. He received his Master and PhD degrees of Computer network from school of computer Science, Northwestern Polytechnical University, in 2000 and 2004, respectively. He had been working for more than 8 years and currently is a associate professor in the school of computer Science at Northwestern Polytechnical University and he had published more than 20 academics paper in journals and conference proceedings. His research interests are network security and highly trusted network.

Wang Xiao received his B.Sc. in Information and Computer Science from Fuzhou University in 2011.Now he is working for his master degree of Computer Science in Northwestern Polytechnical University. His research interests mainly include source track method,network traffic analysis, network provenance and information retrieval.

Wang Min received her master degree of signal processing in school of electronics and information and PhD degree of computer application in school of computer science at Northwestern Polytechnical University in 2000 and 2010 respectively. Her research interests are signal processing and network countermeasures.

Jin Yinghan received his Bachelor degree in School of Automation at Northwestern Polytechnical University in 2010. His research interest is system simulation.